Lexical Classes for structuring the lexicon of a TAG

Benoît Crabbé

Lexical classes for structuring the lexicon of a TAG

Benoit Crabbé
LORIA
BP 239 – Campus Scientifique
F-54506 Vandoeuvre-lès-Nancy
Benoit.Crabbe@loria.fr

Abstract

This paper presents work in progress on a system for structuring the lexicon of a Semantic Tree Adjoining Grammar for French. It focuses on an alternative to lexical rule based structuration of the lexicon, lexical classes: instead of deriving additional lexical structures by means of lexical rules, we show that we can enumerate in a compact way a whole lexicon by combining primitive lexical descriptions.

1 Introduction

This paper presents and motivates linguistically a system for structuring the lexicon of a semantic Tree Adjoining Grammar (TAG). It further assumes some background knowledge of TAG (Joshi and Schabès, 1997).

Throughout the paper, we assign three goals to a system for lexical structuring. (1) Practical, it must in practice ease grammar development and maintenance. (2) Theoretical, it must account for lexical equivalences; in classical systems these equivalences are expressed by means of lexical rules. (3) Genericity, it has not to be specific to our particular linguistic views on syntax.

The organisation is as follows : first, we begin by introducing Semantic TAG, a variant of Feature based TAG (Vijay-Shanker, 1987) where one may express hole semantics with feature structures. Then we present an alternative idea to lexical rules for structuring the lexicon: to do this we first observe two weaknesses of lexical rule based systems (section 3). We then introduce the idea of lexical classes (sections 4-7) designed to avoid problems of rule ordering. Section 8 shows how to overcome the other problem encountered by lexical rules based approaches: the impossibility to guarantee the well formation of the generated lexical units. We finally (Section 9) compare our approach with systems based on similar ideas (Candito, 1999; Xia, 2001).

2 Semantic TAG

We use here Semantic TAG (Gardent and Kallmeyer, 2003), a variant of TAG that allows to express a hole semantics with Feature-Based TAG. On formal grounds, (Gardent and Kallmeyer, 2003) show that semantic TAG can be implemented with Feature-Based TAG, the variant of TAG commonly used by linguists. However it is more convenient to use their idea by considering dual elementary units where a semantic *dimension* is coupled to a syntactic *dimension*. Both are *linked* by feature structure value sharing[1] (Figure 1).

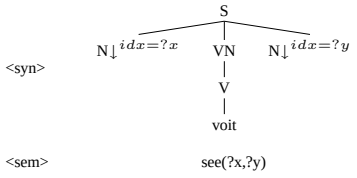


Figure 1: Elementary tree associated with semantics

[1] The trees presented in this paper are highly inspired from the French Grammar of (Abeillé, 2002) where most of the linguistic justifications are given (e.g. no VP). The VN category is a shorthand for *Verbal Nucleus*; its justification is beyond the scope of this paper. The $?x$ and $?y$ denote unification variables. The semantic representation has to be understood as implemented by feature structures. Quantification is not explicitly stated to simplify the reading of the figures.

3 Lexical rules

Traditional ideas underlying lexical rule based systems stem from (Flickinger, 1987). He divides the task of lexical structuring following two axis: a vertical axis where one designs a set of base (canonical) lexical units by means of an inheritance hierarchy, and an horizontal axis meant to express paraphrase relationships between a base lexical unit and its syntactic variations (derived lexical units) by means of lexical rules. This work has been adapted to TAG by (Becker, 1993) and extensively tested by (Prolo, 2002).

Figure 2: Base and derived tree

This paper adresses two critics to lexical rule based systems. The first is that of rule ordering which may lead to quite complicated ordering schemes (see e.g. (Prolo, 2002)) and the second concerns the impossibility to guarantee the well-formedness of the generated structures (see Section 8).

4 Lexical classes

We introduce an idea that overcomes the first critic: that is using lexical classes. Besides the procedural aspects we observe that *in fine* the effect of lexical rules is to associate a finite set of trees to a given lexical item. Our first proposal is straightforward: we suggest to enumerate, one by one, each of those trees that are to be associated to that lexical item.

Such a naive solution sounds meaningless unless we provide (1) a mean to enumerate the list of trees in a compact way and (2) a mechanism that automatically expands the compact description to generate the expanded lexicon. To do this, let us consider two trees expressing two different realisations of a transitive verb (Figure 2). Both trees express a canonical subject and an active verb, while there are two different realisations of the direct object. The tree on the left exhibits a canonical object, while the object is cliticized on the right one. Observe that substracting the object realisation lets two identical trees. Therefore, we can enumerate these two trees with the following statement :

$$SubjAndActiveVerb + (CliticObject \vee CanonicalObject)$$

To output both trees, the expansion mechanism has to take the common description ($SubjAndActiveVerb$) and plug either the $CliticObject$ or the $CanonicalObject$ description (Figure 3).

Though not depicted in Figure 3, subject realisations vary in a similar way as those of the object. Thus we can generalize and describe a wider set of verbal transitive variations with the following description :

$$ActiveAlternation = Subj + ActiveVerb + Object$$

Where $Subj$ (resp. $Object$) has to be understood as a "shortcut" for $SubjCanonical \vee SubjClitic \vee SubjWh \ldots$

More precisely, a "shortcut" such as a $Subj$ or $ActiveAlternation$ has a linguistic relevance. Indeed it describes a set of trees that represent variations equivalent to those that would be expressed by lexical rules, with the difference that there is no special status given to the canonical (base) tree. For this reason we define a *lexical class* as a description that denotes a set of trees that stand in a relation of paraphrase.

As mentioned earlier we want to generate TAGs that support semantic. Therefore we need to add a semantic *dimension* to our syntactic descriptions and link it to the syntactic description. Thanks to lexical classes, trees are grouped in semantically homogeneous sets: trees standing in relation of paraphrase are assumed to share the same semantics. Thus we can add to a lexical class such as $ActiveAlternation$ the semantic representation $see(x, y)$. To perform linking, we have to state that the value of the first predicative argument has to be shared with the value of the subject while the value of the second argument is shared with that of the object whatever the functional realisation is. It entails that each primitive description that represents the realisation of an argument contains a node that is identified with the appropriate function, yielding trees as sketched below:

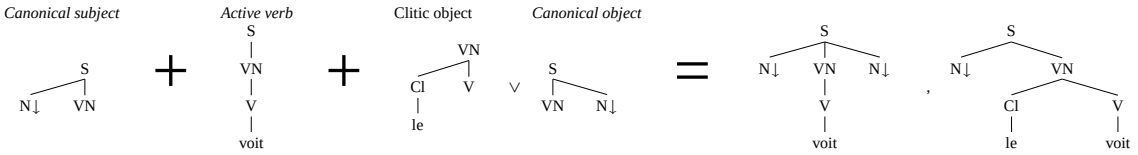
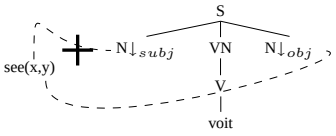


Figure 3: A set of trees as a composition of primitive sub-trees



5 The formal apparatus

To express this idea we need to set up two devices[2]. One, related to the syntactic dimension, for combining sub-trees, the second for performing linking. Semantics is implemented with feature structures.

We begin by introducing a language of tree description, a color based grammar, inspired from foundational work in "model-theoretic syntax". The syntax of the language is made of variables, a connector ($\wedge$), and of a set of binary predicates. The language is interpreted through minimal models that are trees. Variables denote nodes and binary predicates are interpreted as relations over trees such as domination, parent, precedence and adjacency.

In this paper (Figure 4) we use a graphical notation for the language used. Parent is indicated with a solid edge, precedence with $<$ and adjacency with «[3].

Moreover, each variable is associated to a property, its color. The color property has three values: white ($\circ$), black ($\bullet$) and red (ø). Colors may be combined. A white node combined with a black node yields a black node, a white node combined with a white node yields a white node, a black node cannot be combined with a black node, and a red node cannot be combined at all. With this property, a model is valid if (1) it is a tree and (2) every node in the tree is either red or black.

The intuition behind this, is that black nodes are those nodes where one can attach another partial tree, that white nodes are the nodes that need to be attached to another partial tree and that red nodes are the nodes where nothing can be attached.

Eventually, as commonly done in computational linguistics, each variable may be associated to an attribute value matrix. Features are combined by unification. Therefore models are valid if for every node feature structure unification succeeds.

Lexical information is expressed within a system of macros. A macro (or class[4]) is made of a syntactic dimension where partial trees are specified, and/or a semantic dimension where the semantic representation is specified by means of feature structures, and/or a specification of its subclasses. The sub-class specification is a conjunction and/or a disjunction of sub-classes.

To perform linking, we need to allow sharing of information between the syntactic dimension where trees are described and the semantic dimension. To do this, we allow the value of a feature structure associated to a tree node to be coindexed with a value of the feature structure implementing the semantics.

Feature structure sharing between dimensions is made possible by an additional system of feature structures called *interfaces*. Values of interfaces features may be shared with values of features from the syntactic or the semantic content. Interfaces are thus meant to allow a given class to grant access to some of its feature values to any other class.

The expansion (or evaluation) of a class A consists of a recursive undeterministic expansion of its sub-classes. Moreover, this process combines (1) the syntactic dimension (2) the semantic dimension and (3) the interfaces of an instance of A and of instances of all of its subclasses. Syntactic descriptions are conjoined, semantic descriptions

[2]The formal system overviewed here is described in details by (Duchier, 2003).

[3]Two nodes x and y are adjacent iff they are siblings and there is no node z such that $x < z < y$.

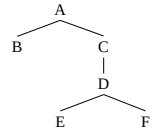
[4]We use the term class because those macros can be interpreted as a multiple inheritance hierarchy.

and interfaces are unified[5]. For instance, with the following class definitions,

`class` **`A`** `= B ∨ (C ∧ D)`

`class` **`D`** `= E ∨ F`

stating that class A (resp. D) is subclassed by either class B (resp. E) or classes C and D (resp. F). According to this definition, an expansion schema for class A is:



Thus the system builds lexical units by combining information from A and B, from A, C, D and E and finally from A, C, D and F.

An intuitive way to understand the process of expansion is to think in terms of macro expansion. However that comparison is not perfectly correct, because the expansion mechanism described here combines instances of classes. So it makes sense to combine an instance of class A with another instance of A. That is important, if one wants to express, elementary trees in which there are two prepositional phrases, or with a double extraction.

6 An example

We show how to express the example given informally in section 4 with the actual system.

We begin by the syntactic dimension. An active alternation is defined as:

`class` **`ActiveAlt`** `=`

`Subj ∧ ActiveVerb ∧ Object`

Classes embedding syntactic functions are defined as:

`class` **`Subj`** `= canonical subject ∨ Wh subj ∨ clitic subj`

`class` **`Object`** `= canonical object ∨ wh obj ∨ clitic object`

Some classes describing syntactic realisations of functions are described in Figure 4. For each class, its name is noted in italics, the syntactic content is noted with the graphical notation given in section 5 and the interface features are noted within square brackets.

We now turn to the semantic dimension. The semantics of *see* is described with:

`class` **`SemanticsOfSee`** `= <sem>{see(?X,?Y)} [ARG0 = ?X, ARG1 = ?Y]`

where we specify a semantic content with the feature structure see(?X,?Y). ?X and ?Y are unification variables coindexed with the interface features which are again noted within square brackets.

And finally, linking is expressed within the class:

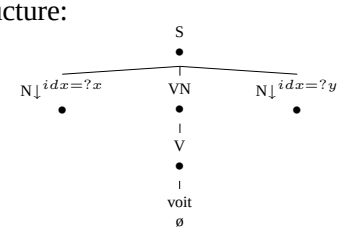
`class` **`LexEntrySee`** `=`

`ActiveAlt[Subj = ?X, Obj = ?Y]`

`∧ SemanticsOfSee[ARG0=?X, ARG1=?Y]`

Among the alternative lexical structures generated by expanding the class *LexEntrySee*, we choose to show in details how the lexical unit depicted in Figure 1 is generated. It is produced by combining information from the following classes: *LexEntrySee–SemanticsOfSee–ActiveAlt–Subj–canonicalSubject–ActiveVerb–Object–canonicalObject*.

The syntactic content is specified in the classes *canonical Subject – ActiveVerb–canonical object*. Their combination thus outputs the following structure:



The semantic content is specified only in the class *SemanticsOfSee* and is thus $see(?x, ?y)$.

The interfaces attached to *ActiveAlt, SemanticsOfSee, canonical subject* and *canonical object* are unified which results in the following feature structure:

$$\begin{bmatrix} \text{SUBJ} & ?X \\ \text{OBJ} & ?Y \\ \text{ARG0} & ?X \\ \text{ARG1} & ?Y \end{bmatrix}$$

As the values of SUBJ and OBJ are shared with the appropriate feature values in the syntactic content (Figure 4) and the values of ARG0 and ARG1 are shared with the semantic content[6], we end up with the lexical structure depicted in Figure 1. Stated differently, linking between syntax and semantics is expressed indirectly thanks to the interfaces.

[5] However we do not detail additional combination constraints on interfaces as stated in (Duchier, 2003).

[6] This information is specifi ed in the class *SemanticsOfSee*.

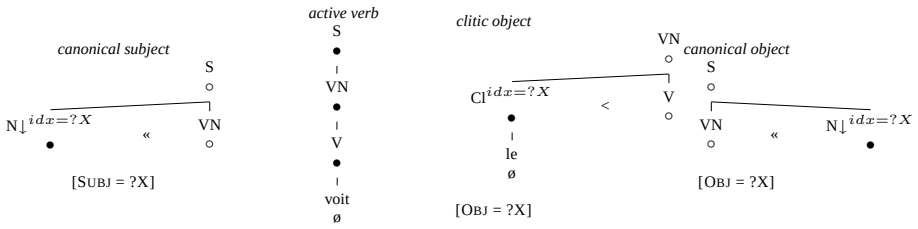


Figure 4: Classes expressing realisations of syntactic sub-trees

7 Erasing: a counter example ?

In the literature, it is often argued that the lexicon has to be managed by a procedural system such as lexical rules because of erasing (Evans et al., 2000). A well known case of erasing is the *agentless passive* where an "initial subject" argument is said to be dropped. Generally speaking, this argument is wrong. It applies only to grammatical systems build upon a single level of representation (like HPSG). The version of TAG we use is interpreted as a two-level grammatical system (like LFG), made of a syntactic and a semantic dimension, where there is no need to drop or erase arguments. Instead we do not link the semantic argument with a syntactic realisation. Thus we can add to the lexical entry of see the agentless passive alternation in which ARG0 is not linked :

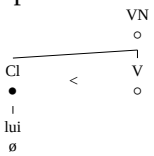
```
class LexEntrySee = (
ActiveAlt [Subj = ?X, Obj = ?Y]
∨ AgtlessPass[Subj =?Y]
)
∧ SemanticsOfSee[ARG0=?X, ARG1=?Y]
```

8 Lexical principles

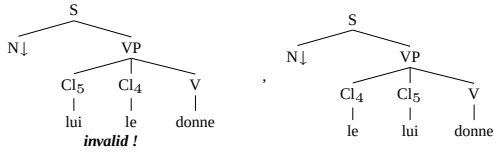
An important point concerns the guarantee of well-formedness of the output trees. We use lexical principles to account for "surface" interactions between independently described phenomenons. For instance in a French grammar, one may express clitics as coanchors of elementary trees. In French (and Romance languages in general) clitics are ordered according to a fixed order[7]. As other frameworks, our system also faces the problem of clitic ordering. To illustrate this let us consider the following macros:

```
class DitransitiveActive = Subj ∧
ActiveVerb ∧ Object ∧ IndirectObject
class IndirectObject = canonicalIndir ∨
cliticIndir ∨...
```

We further assume that *cliticIndir* contains a description such as:



Among the expansions of $DitransitiveActive$ we get a case where the syntactic content of $cliticIndir$ and $cliticObject$ (Figure 4) are combined, thus leading to two solutions such as :



The left solution violates the ordering of clitics. To remedy this, we specify a lexical principle of clitic ordering ensuring the well formation of every tree output by the system. As illustrated in the picture above, nodes of category clitic (Cl) are further associated to an additional property, their rank[8] (noted in subscript). The principle just puts an additional condition on the class of target models: if two sibling nodes have the rank property then they have to be ordered according to their rank. In our present case, the principle says that the clitic of rank 4 has to precede a clitic of rank 5, thus the structure on the left is not a model anymore[9].

The idea of stating principles of this kind could be generalized to express stronger principles such as those suggested by (Frank, 2002).

[7] See e.g. (Perlmutter, 1970). Besides clitics one may find other "surface constraints" which are related to subject inversion and wh-island representation.

[8] See e.g. (Perlmutter, 1970) for further details.

[9] Note that for explanatory purposes, the principle remains simplified: the ordering is different when the verb is imperative.

9 Related work

Alternative ideas to lexical rule based approaches for managing the lexicon of a TAG have been first introduced by (Candito, 1999) and (Xia, 2001). These works are close to ours. According to our initial goals our gains are theoretical, practical and address also genericity.

On theorical grounds, we draw an explicit distinction between the two axis of (Flickinger, 1987) (vertical vs horizontal). Doing this enables to make a comparison between our ideas and lexical rule based systems.

On practical grounds, we remain declarative, section 7 shows in particular how to overcome the "erasing" argument; the solution we adopt avoids the introduction of obscure devices for expressing erasing as does (Candito, 1999) or to make an artificial split between two kinds of lexical rules such as does (Xia, 2001)[10].

Regarding genericity, our system is not restricted to 3 levels[11]. The organisation of the lexicon in 3 levels suggested by (Candito, 1999) can still be used as a methodology to ease grammar development, but it is not mandatory. One could use our system for e.g. representing lexical alternations in a framework close to (Levin, 1993)[12].

Finally we put forward the use of lexical principles to ensure correctness of the output trees, which are not tackled by any system except (Candito, 1999). These principles are not proper to our approach: similar principles should solve problems of artificial multiplication of lexical rules as observed by (Prolo, 2002).

10 Conclusion and perspectives

We propose and motivate a system for lexical structuring which is declarative. The full-formal aspects and the implementation are described in (Duchier, 2003). This paper describes only the so-called horizontal axis (Flickinger, 1987), comparable to lexical rules. It has still to be augmented in order to describe a vertical axis, similar to the inheritance hierarchy of (Flickinger, 1987) where one could factor out the primitive tree descriptions described in this paper.

[10](Xia, 2001) makes an implicit distinction between "monotonic lexical rules" (building blocks) and non monotonic ones (Lexical redistribution rules) which is hard to justify on linguistical grounds. (Gaiffe et al., 2002) who generalizes (Candito, 1999) by allowing an unrestricted number of dimensions does not allow to express erasing at all.

[11]Our dimensions should not be confused with those of (Candito, 1999).

[12]See (Crabbé, 2003).

References

Anne Abeillé. 2002. *Une grammaire d'arbres adjoints pour le français*. Editions du CNRS, Paris.

Tilman Becker. 1993. *HyTAG : A new Type of Tree Adjoining Grammars for Hybrid Syntactic Representation of Free Word Order Language*. Ph.D. thesis, Universität des Saarlandes.

Marie-Hélène Candito. 1999. *Organisation Modulaire et Paramétrable de Grammaires Electroniques Lexicalisées*. Ph.D. thesis, Université de Paris 7.

Benoit Crabbé. 2003. Alternations, monotonicity and the lexicon: an application to factorising information in a tree adjoining grammar. In *Proceedings of ESSLLI'03*.

Denys Duchier. 2003. A metagrammatical formalism for lexicalized tags. In *Prospects and advances in the syntax/semantics interface*.

Roger Evans, Gerald Gazdar, and David Weir. 2000. 'lexical rules' are just lexical rules. In Anne Abeillé and Owen Rambow, editors, *Tree Adjoining Grammars. Formalisms, Linguistic Analysis and Processing*. CSLI, Stanford.

Daniel Flickinger. 1987. *Lexical Rules in the Hierarchical Lexicon*. Ph.D. thesis, Stanford University.

Robert Frank. 2002. *Phrase Structure Composition and Syntactic Dependencies*. MIT Press, Boston.

B. Gaiffe, B. Crabbé, and A. Roussanaly. 2002. A new metagrammar compiler. *Proceedings of TAG+6*.

Claire Gardent and Laura Kallmeyer. 2003. Semantic construction in feature-based tree adjoining grammar. *Proceedings of the 10th conference of the European Chapter of the Association for Computational Linguistics*.

Aravind K. Joshi and Yves Schabès. 1997. Tree adjoining grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*. Springer Verlag, Berlin.

Beth Levin. 1993. *English Verb Classes and Alternations*. The University of Chicago Press.

David Perlmutter. 1970. Surface structure constraints in syntax. *Linguistic Inquiry*, 1:187–255.

Carlos Prolo. 2002. Systematic grammar development int the XTAG project. *Proceedings of COLING'02*.

K. Vijay-Shanker. 1987. *A study of Tree Adjoining Grammar*. Ph.D. thesis, Department. of computer and information science, University of Pennsylvania.

Fei Xia. 2001. *Automatic Grammar Generation from two Different Perspectives*. Ph.D. thesis, University of Pennsylvania.